Bachelor's Thesis

A Comparison of Deep Learning Systems for Phone Recognition

Author:
Samantha Tureski

Supervisors:
Prof. Kurt Eberle
Dr. Fabian Tomaschek

Seminar für Sprachwissenschaft
Eberhard-Karls-Universität Tübingen

December 2018

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst, keine anderen als die angegebenen Hilfsmittel und Quellen benutzt, alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe und dass die Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist und dass die Arbeit weder vollständig noch in wesentlichen Teilen bereits veröffentlicht wurde sowie dass das in Dateiform eingereichte Exemplar mit den eingereichten gebundenen Exemplaren übereinstimmt.

Tübingen, December 13, 2018

Samantha Tureski

Contents

1 Introduction **1**

2 Motivation and Related Work **3**
2.1 Motivation 3
2.2 Related Work 6

3 Model Architectures and Experiments **11**
3.1 The Dataset 11
3.2 Audio processing 12
3.3 The Models 13
3.3.1 The base model 13
3.3.2 Transfer learning 15
3.3.3 CNN Features in Conjunction with a Support Vector Machine 16
3.3.4 CNN Features in Conjunction with a Gated Recurrent Unit . 17

4 Results and Discussion **17**
4.1 Transfer learning results 18
4.2 CNN-SVM Results 20
4.3 CNN-GRU Results 22

5 Conclusion **23**

6 Acknowledgements **23**

References **23**

Abstract

Deep neural networks, which employ nonlinear iterative frameworks to recognize patterns, have emerged in recent years as an ideal basis for automated speech recognition (ASR) systems. While the research community has proposed numerous models to generate text from speech, error rates for individual phone classification from raw speech data remain relatively high. This thesis seeks to investigate and improve upon currently available deep learning techniques for phone recognition. Beyond providing the motivation for this project, this paper describes a broad range of promising research in the field. Several self-implemented solutions, which had not yet been applied to phone recognition in the existing literature, are also presented. These CNN-based models trained on MFCC features incorporate Gated Recurrent Units and two types of transfer learning. The outcomes of the deep learning models used approach state-of-the-art accuracy on consumer-grade resources for this complex, multi-label problem.

1 Introduction

As the use of technology becomes increasingly common in daily life, ease of interaction with computers has become a necessity. Since speech is one of the most natural forms of human communication, it is expected that future computer applications will require researchers to seamlessly transcend the language barrier between humans and machines. Already, commercially available automatic speech recognition (ASR) systems have been released to aid in various tasks, such as hands-free typing, real-time translation, and providing virtual assistants like Apple's Siri and Amazon's Alexa. Research, as well, is poised to benefit from advances in ASR. In speech and audio data, there lies a wealth of information that, when transcribed accurately, may rival the information that exists in digital text formats. Automated speech recognition (ASR) is the task of converting speech from a recorded audio signal to text (Gruhn & Hamerich, 2011). Many traditional and successful ASR systems have focused on decreasing word-error rate by relying on word-based models. In these systems, a set vocabulary or n-gram model restrict the possible transcriptions for an utterance. In the case of this paper, I choose to explore ASR concerned with evaluating speech at the phone level. A phone is defined as "an absolute speech segment that possesses distinct physical properties and serves as the basic unit of phonetic speech analysis" (Crystal, 2011), whereas a phoneme is a contrastive sound that recognizes functional differences within a language (Clark & Yallop, 1994). For the purposes of this paper, I refer to phones, though language-specific research has used the word phoneme (such as the German annotation tool developed by Rapp (1995)).

The motivations for identifying speech on a phone level are threefold. First, there was a need for a phone recognition system in the project that I worked for as a research assistant (see Section 2.1). Second, the problem is crucial to the advancement of ASR research: speech signals contain an enormous variance due to speaker age, gender, language, dialect, background noise, and even size of articulators. It is crucial that an ASR model is flexible enough to account for these differences, always guessing, for example, the word "cat" from thousands of different English speakers. In addition, systems that recognize speech by matching audio with words from a set vocabulary will not be able to recognize out-of-vocabulary (OOV) terms. Highly accurate ASR solutions require a fine-grained way to accommodate the ever-changing and diverse nature of vocabulary and usage. Phone recognition can fill in the gaps where word-based speech recognition fails, or can serve as the basis for completely vocabulary independent (VI) systems. Classification of phones has proven essential in applications such as language recognition, speaker identification, and keyword detection. As Lamel & Gauvain (1993) note, research focused on simple

word recognition can obscure issues that may be occurring on the phone level. Lastly, with the best accuracy rates hovering below 80% (Lopes & Perdigao, 2011), computer recognition of phones lag behind similar tasks such as digit recognition (99.6%, (Li, 2008)) or face recognition from the standard FaceNet database (99.63%, (Schroff et al., 2015)). Thus, there remains a considerable amount of interest within the research community to improve accuracy rates for this important task. Automated speech recognition (ASR) systems, both phone and word-based, have traditionally relied upon Hidden Markov Models (HMMs) and Viterbi decoders. More recently, deep learning approaches, particularly those involving recurrent neural networks (RNNs) and convolutional neural networks (CNNs) have been demonstrated to produce state-of-the-art results for the same tasks (Amodei et al., 2016). The complex mathematics behind deep neural networks allow them to "learn" phone or word labels from raw speech data, offering an attractive solution to a problem that once required more cumbersome, human-engineered pipelines. The flexible nature of neural networks also allow them to account for the many different variables among speakers.

Deep learning with neural networks, however, is not without its disadvantages. As a supervised learning approach, it requires vast amounts of hand-labeled data, which is labor-intensive and often unavailable for languages that are not as widely spoken as English. In the case of phone recognition, a model must be trained on a dataset consisting of audio clips of human speech with each phone correctly labeled- a tedious job for even an experienced linguist. Another disadvantage of neural networks is that they depend on thousands of computations to continuously update their parameters, which can necessitate a considerable amount of compute resources and expensive hardware.

In this thesis, I am particularly interested in studying and building computationally efficient models that can be run on my own computer's specifications (Intel HD Graphics 6000 integrated GPU, 8 GB of RAM, and a Intel HD 6000 1536 MB graphics card) and that could be reproduced by those without access to professional resources. Research in the field of vocabulary-based ASR has identified a number of ways by which model architectures can be modified to return state-of-the-art results on consumer-grade GPUs. These include reducing dependence on context, employing convolutional neural networks as feature extractors, implementing a technique called transfer learning, and using gated recurrent units (GRUs) instead of long-short term memory units (LSTMs) to capture long-term dependencies ((Wang & Zheng, 2015), (Xu et al., 2017)). Building on the existing literature, this thesis serves as a preliminary investigation into three different deep learning systems that had not before been applied to problem of automatic recognition of phones.

All three of my experiments utilize German speech data. Audio signals are

reduced to features based on Mel-frequency cepstral coefficients (MFCCs) and their log coefficients, known as delta and delta delta features. While teams like Amodei et al. (2016) encounter good results with complicated recurrent neural network (RNN) hybrid architectures, I use a simple convolutional neural network (CNN) architecture as the basis for all three experiments. The basic architecture is modified according to the computation-reducing techniques mentioned above. Phone labels are estimated by each model based on phonetic features extracted from the speech signal.

Through its recursive structure, a gated recurrent unit is able to account for information over a time series rather than just at discrete moments. For this reason, my CNN-GRU model is expected to outperform the other models, which do not carry this contextual advantage. The transfer learning model works by initializing a model with pre-trained rather than the random weights typically generated for CNNs, so I also expect the transfer learning models to yield better results than the base model. Additionally, I expect for there to be a correlation in accuracy depending on at which layers they have pre-trained weights. Another form of transfer learning pipes high-dimensional features from a CNN into a Support Vector Machine. The prognosis for this type of model is unclear, though it is an interesting idea which has not been heavily researched. By efficiently leveraging these techniques, all of the models are expected to achieve suitable results for phone recognition.

This paper is structured as follows: Section 2 explains the motivation for this project and provides an overview of existing approaches to ASR, while Section 3 details the implementation of the three deep learning experiments devised for this thesis, as well as the reasoning behind them. Section 4 encompasses the evaluation and comparison of all of the models. Finally, Section 5 presents a summary and conclusion of the project.

2 Motivation and Related Work

2.1 Motivation

The original short-term goal of this thesis emerged when I worked as a research assistant at the faculty of Quantitative Linguistics at the University of Tübingen from December 2017 until July 2018. One of my assignments was to create an automatic labelling tool for short speech clips from the Karl Eberhards Corpus (KEC) (Arnold & Tomaschek, 2017). The Karl Eberhards Corpus, in its current release, offers manually corrected word boundaries and automatically aligned segment (i.e. phone) boundaries. Research assistants are also currently in the process of inspecting segment boundaries. The corpus' documentation, however, lacks proper segment labels within shorter ut-

terances. Full alignment and annotation of the Karl Eberhards Corpus remains an important goal because, relative to English, German does not have the same multitude of data required to advance research in supervised ASR techniques. The corpus, which was first released by the University of Tübingen in 2017, currently consists of 35 hours of spontaneous dialog in German. Spontaneously elicited conversations in telephone-style format offer an obvious environment to facilitate natural sounding speech and turn taking between acquainted parties, and had not been previously offered in the well-known GECO Corpus for German (Schweitzer & Lewandowski, 2013). Word and segment boundaries have been automatically corrected by Stuttgart Aligner, an audio segmenter that was developed at the University of Stuttgart in 1995 (Rapp, 1995). The Stuttgart Aligner discovers word and phoneme boundaries using Hidden Markov Models and a Viterbi decoder. Given German word-level annotations for an audio file, the Aligner automatically transcribes the words into their phonemes via lexicon lookup or rule-based conversion, then predicts segment boundaries and fills in the phoneme labels. The paper behind the Aligner notes one considerable downfall, however: "for a 20 ms accuracy (two frames) the [alignment] procedure reaches 76.82...one fourth of the automatically found segment boundaries mismatch the manual labelling by 20 ms or more. . . [F]or fine phonetic labelling for which a time resolution under 10 to 20 ms is required other alignment tools should be conceptualised." This means that the Aligner does well in finding the general boundaries for words, which tend to have durations greater than 20 ms in the KEC. But for phones and quickly-spoken words, more careful manual boundary correction is necessary. Phones within quickly-spoken words pose an even greater challenge, because even human annotators like myself and colleagues were sometimes unable to distinguish the contents of the sound clip. Another tool had to be developed in order for the Karl Eberhards Corpus to achieve full, accurate documentation in this area. Tasked with filling in this gap are the experimental models that are described in the current paper.

The KEC is released with files that can be viewed using the popular phonetic analysis software Praat, with labels and boundaries conveniently displayed under their corresponding audio segments. As can be seen in Figure 2.1, short words in the corpus are marked with the label 'ASF', followed by the possible word label. Potential segments within these words carry the label 'AFS' and an identification marker. In my time as a research assistant, I spent hours inspecting and correcting phone boundaries for the corpus. As a result, I became exceedingly familiar with the contents of the short, 'ASF'-labeled audio clips. Most typically, the utterances contained stop words or reduced versions of them (common tokens like 'ja', 'also', 'das', 'dann', ''ne'), filler words ('aeh'), and sometimes unintelligible sounds. As stated above, the

current pipeline for producing corpus documentation lacks the functionality to automatically transcribe phones contained in these short pieces of speech. The main idea of the current project is to isolate the short sound clips and label the phone(s) within those clips. To efficiently label the existing parts of the corpus, and to aid researchers adding more data to the KEC in coming years, an in-house phoneme recognition tool is a necessary addition to the Aligner. However, this thesis does not chronicle the development of the full tool, but rather the extensive research into phone recognition that comes before it. Because the possible word labels and word boundaries already exist for the short words, I choose to focus on improving recognition of the individual phones, rather than developing an end-to-end system that guesses entire words from speech.

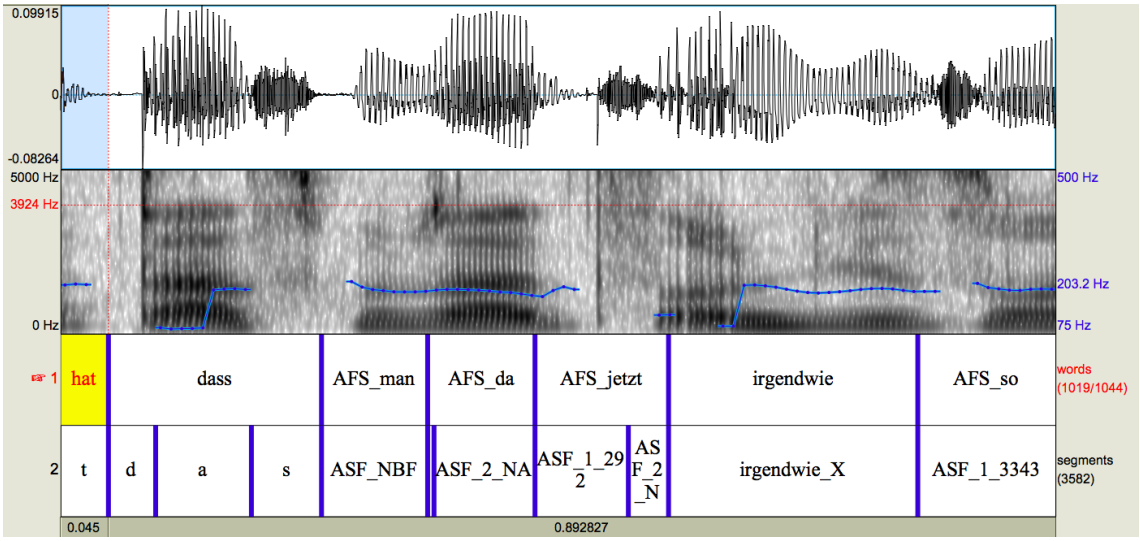

Figure 2.1: As is clear under 'man' in the picture, individual phone boundaries have not always been found within a word. If this is the case, the future phone recognition tool can predict phone labels at uniform interval. Otherwise, the labels for the predetermined segments will be predicted.

Thanks to the Aligner, the corpus frequently contains the millisecond boundaries that denote the estimated beginning and end of an unknown segment. A memory-efficient approach, then, would be to train models on only these audio clips, with minimal context added. While the ASR industry is currently leaning away from phonemes as independent "beads on a string" model of phones (Ostendorf, 1999), restricted context or context-free phone recognition has has shown promising results in a number of papers ((Young, 1992),(Golipour & O'Shaughnessy, 2009)). In three out of four of my models, I choose to use only small amounts of left and right contextual information. Restricting audio context for phone recognition has several advantages. First, it requires fewer computing resources and trains faster. Next, it is ideal for data that is broken into discrete segments of audio, whether due to a docu-

mentation format as in the KEC's case, or for when audio data is missing or unreliable (Cooke et al., 2001) Lastly, observing phones in their own right is an interesting research question in itself. Recent research has suggested that phones themselves may not exist as well-defined units but are only meaningful to the listener relative to their context within a much longer utterance. It has been remarked upon, for instance, that the difference between postvocalic *p* and *b* is often indicated by the duration of the vowel (Raphael, 1972). Certain linguists believe that phones are a purely theoretical structure that only loosely describe actual speech, and do not, in fact, proceed from the written form of the word. Indeed, reduced forms of words in spontaneous speech can complicate phone-based speech recognition. In a study of American English speakers, a little over 20% of words were realized in speech with at least one phone missing (Johnson, 2004). Furthermore, many humans cannot understand short sound clips out of context, as we found while correcting the Karl Eberhards Corpus. Ostendorf (1999) reports that that phone-based features may be inferior to a broader range of features required for speech comprehension. Successful implementation and improvement upon reduced-context phone recognition systems, then, would reinforce the traditional view of phones that has been challenged in recent years. The contribution of this thesis is to test a number of known techniques to create an moderately accurate phone classification engine in a reasonable period of time, given student experience and resources. Groundbreaking innovations in the field are left to professional research teams.

2.2 Related Work

Phone recognition

Phone recognition is the task of, given input speech X, finding the most likely sequence of Phones Ph^*within it. This search problem can be modeled by the equation:

$$Ph^* = argmaxP(Ph|X)$$

(Lopes & Perdigao, 2011)

Research into phone recognition attempts to optimize accuracy scores for this task. Ludwig Maximilian University of Munich, which has released a number of phone-related tools such as MAUS (Kipp et al., 1997), defines the standard equation for phone accuracy as:

$$Accuracy = \frac{N - D - S}{N}$$

Where N is the total labels in the speech clip under consideration, D is the number of phone deletions, and S is the number of substitutions. Lopes & Perdigao (2011) present a slightly different metric that also penalizes for the number of erroneously inserted phones I. The past thirty years has seen a growing body of research into the task of phone recognition. The English language TIMIT dataset is frequently used due to its hand-checked phone annotation and moderate size (Zue et al., 1990). Phone recognition experiments from the 1980s and 90s frequently rely on a traditional Bayesian approach. These models, which encompass Hidden Markov Models (HMMs) and GMMs, attempt to fit predictions into a learned probability distribution. Given a set of acoustic features within a speech signal X, a Bayesian-style model will aim to predict the most likely phone label Ph on the basis of the formula $P(X|Ph)P(Ph)$. In their 1989 benchmark paper, Lee and Hon achieve 66.08% accuracy using a Hidden Markov Model on the TIMIT dataset. They pipe diphone (i.e. a phone and the one before it) probabilities into a bigram language model. A number of other research teams have gone on to implement their own forms of phone-recognizing HMMs. Young (1992) and Lamel & Gauvain (1993), for instance, achieve respective accuracies of 59.9% and 72.9%. As Mohamed et al. (2011) note, however, Hidden Markov Models may be fundamentally ill-suited to the task of ASR. The first-order Markov chain assumption that each state S_t at time t depends only on S_{t-1}, and that observations at different time steps are conditionally independent. Such assumptions are unrealistic when considering the complicated interactions within speech signals.

With the advent of greater computing power in the 1990s, deep learning emerged as a competitor to traditional Bayesian approaches. Deep architectures, including neural networks and deep belief networks, seek to closely model data through layered sets containing thousands of parameters. The term "deep" refers to the nature of the unobservable layers through which data is transformed. Deep models continuously update these parameters in order to reduce the discrepancy between estimated and predicted output. Thus, unlike earlier statistical methods, deep approaches generate a class distribution from the data rather than assume a pre-existing distribution. Robinson (1994) is one of the first to apply deep learning to phone recognition using the TIMIT data set. Just as in my project, Robinson implements the network after the speech has been decoded by an HMM-based decoder. The 75% accuracy score achieved remains competitive even to this day. The current leader in performance for the phone recognition task is also the product of a deep learning model: the monophone deep belief networks developed by Mohamed et al. (2009) recognize phones at 79% accuracy. I found these methods promising and, as I was familiar with neural networks from my university studies, decided

to pursue this route for the project. I sought to find network architectures that most faithfully model acoustic data. After some research, I learned that speech data can be represented as waveform images or as condensed numerical vectors that, if properly interpreted and graphed, mimic the original waveform. Convolutional neural networks, then, are a natural choice, as they perform best on image data. The next few subsections will provide a brief theoretical overview into the components of the models described in the experiments of Section 3.3.

Convolutional neural networks

Convolutional Neural Networks (CNNs) learn to classify input data by passing it through a series of layers. Simple CNN stack layers in a sequence typically similar to the following:

1. Convolutional layers employ a filter that slides through the width and height of an input feature vector, and computes the dot product of the input's region and the weight learning parameters. The resulting 2-dimensional activation map consists of responses of the filter at each region.
2. A layer with a Rectified Linear Unit (or ReLU) applies an activation function $F(x) = max(0, x)$. Any negative values are changed to zero to increase computational efficiency and introduce non-linearity.
3. A pooling layer provides dimensionality reduction, distilling the most notable features into a smaller feature vector. This helps to reduce memory usage and overfitting (i.e. when a model conforms too closely to its training data), while maintaining rough spatial information. In a max pooling, for example, a small window is passed over the feature matrix at non-overlapping intervals. The highest value within the window is selected to be passed forward to the next layer, while the other values are discarded.
4. A fully connected layer, with connections to all activations in the previous layer, will compute scores for each of the possible classes. This last dense layer is the same kind found in a typical neural network. The class with the highest score is the one the image or feature vector most likely depicts.

   Abdel-Hamid et al. (2012) describe in depth how CNNs faithfully model features from an acoustic signal. Their CNN-HMM model shows an 11 error reduction compared to speech recognition using a typical neural network, achieving a 79.9% accuracy score. For the sake of brevity in this bachelor's thesis, I have chosen to omit a full introduction to neural networks. A more comprehensive introduction can be found in Haykin (1994).

Transfer learning

Since CNN's appeared promising, I chose to delve further into efficiency increasing modifications to the main architecture. The technique of transfer learning does exactly this by allowing one model to reuse the experience learned by another model. For speech recognition, this is an important idea because it harnesses the statistical patterns that are common at the most basic levels within speech across speakers, environments, and languages. In image processing, for instance, a model in its early stages will almost always learn to identify color blobs or general line orientations (Yosinski et al., 2014). Instead of initializing a neural network with random weights, as is typical, a model with pre-trained weights can more quickly discern higher-level features within training data, reducing GPU usage and training time. While there are many categories of transfer learning, I choose to focus of a simple form called model adaptation (Wang & Zheng, 2015). Model adaptation, in which an existing model is adapted to handle a change in data distribution, requires that "both features and label spaces are the same for the target domains, and the models are the same." I want to use a CNN model and transfer weights from another CNN. A slightly different form of transfer learning is used successfully in end-to-end (i.e. speech to full contents of utterances) German speech recognition by Kunze et al. (2017). The same model is used, but trained on datasets from separate languages. Their approach leverages weights from a neural network trained on a high-volume English dataset (Librivox, with 1000 hours of read speech) to be used in the first layers of a second model trained on a German dataset. This was interesting, but I desired to find proof that transfer learning would work for a simple CNN architecture. Yosinski et al. (2014) provide an analysis of transfer learning for CNNs. They test performance of networks of k layers that have been given transfer weights for all layers an $k-1$, for different values of k. They also test the effect of freezing weights for a given layer (rendering them unable to be trained during backpropagation). Their research shows that, even when a dataset (in this case, Imagenet) is split in half, the weights for a model trained on one half will generate up to a 2.1% mean performance boost when transferred to the model trained on the second half. Performance improved even as higher level layers were initialized with transfer weights, as long as they were left trainable. Transfer learning appeared to be an excellent idea. I wanted to test other efficient approaches. I learned that the CNN can discover powerful latent associations as through its manipulation of images (or image-like) data. Sharif Razavian et al. (2014)show that features extracted out of a CNN's final layer can be used in different classifiers rather than being classified

by the fully connected layer. This team reports state-of-the-art performance for image recognition when passing these CNN-created features through a linear SVM (Support Vector Machine). I decided to emulate this alternate transfer learning technique by applying it to my own task. Here I provide a short conceptual explanation for SVMs:

Support Vector Machines

The goal of the SVM algorithm is to find a classifier (linear or nonlinear) that optimizes the separation of classes in a multidimensional vector space. A suitable classifier, for example, will have the largest distance (i.e. margin) from data points within any of the classes and misclassifies as few data points as possible. SVMs are easily implementable through the Scikit library for python(Pedregosa et al., 2011). A simple plot of the algorithm from Scikit learn documentation of a support vector machine can be seen in Figure 2.2.

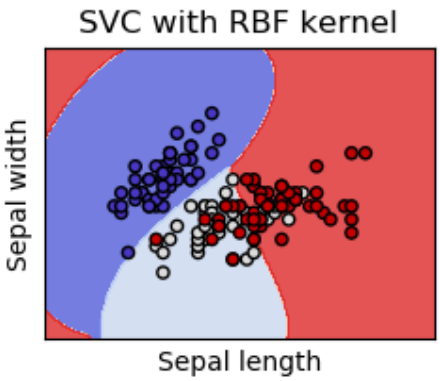

Figure 2.2: Here, the separation of three classes of different flowers by an SVM is flattened onto on a 2D axis for visualization. *Source: https://scikit-learn.org/stable/modules/svm.html*

Users of the SVM function can set a custom regularization parameter, which constrains margin size versus the number of misclassified data points, as well as the γ parameter, which determines the influence of any one data point on margin placement.

Gated Recurrent Units

A second method for harnessing CNN-discovered features is to use them in a way that will maintain long-range temporal information. With my colleague and research partner Ryan Callihan I worked on the interesting and poorly researched combination of piping CNN features into a Gated Recurrent Unit, also known as a GRU cell. Gated Recurrent Units are a type of deep recurrent neural network. What sets them apart from the more popular Long Short Term Memory (LSTM) architecture is that they are more computationally efficient. A Gated Recurrent Unit consists

of a reset gate r, and an update gate z. The reset gate combine the new input with the previous memory, and the update gate defines how much of the previous memory will stay in the network. Without entering into the finer technical details, a GRU cell can be conceptualized in Figure 2.3.

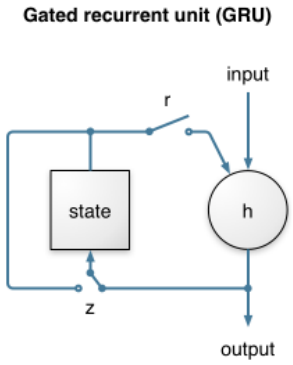

Figure 2.3: In this image, the gates z and r are clearly depicted, and can "open" as necessary to allow information from the previous state into the current network h. *Source: https://people.xiph.org/~jm/demo/rnnoise/*

Further theoretical information information about the CNN-GRU hybrid model for speech recognition can be found in Xu et al. (2017).

3 Model Architectures and Experiments

3.1 The Dataset

Originally I wanted to use the Karl Eberhards Corpus as my dataset. My colleagues and I had spent numerous hours correcting automatically-issued phoneme labels and segment boundaries. This was partly to ensure an accurately labelled release of the corpus, and partly in the hope that the non-'ASF'-labeled KEC data (i.e. the segments with the known phone labels), being most similar to the unrecognized sound clips, could be the best data to train any models. The 'ASF' marking, in theory, could be used as a tag to extract and test only these segments in a phoneme recognition tool for test purposes. However, there exist no gold standard labels for the 'ASF' phones, so they could not be used as a test set, which, for calculation of accuracy and other measures, require pre-known labels. This fact, in addition to the relatively small amount of the data that do have labels, make the KEC inappropriate for the purposes of building a model. Instead, the Kiel Corpus was chosen as the next best option. It was ideal for my purposes because it is a German language corpus that is already split into time segments, and boasts an

extensive annotation scheme for phones. The Kiel Read Corpus, first released in 1996, speech modified and augmented SAMPA (Kohler, 1996). It contains 5 hours and 41 minutes of annotated read speech in standard German sampled at a rate of 16 kHz, along with the corresponding text files. The text files contain phone annotation along with their start times, in milliseconds. (There exists a larger Kiel Corpus of Spontaneous Speech, which may have been more similar to the data KEC, though it is not publicly available and expensive to purchase.) The audio files are annotated not only for their canonical German pronunciation, but also for phonetic variance within the individual instances of pronunciation. Prosody is also annotated; however, this information was not used in this project. The Kiel Corpus is composed of an intricately annotated phone character set, while the KEC is annotated in standard German SAMPA, a well-known system in which each International Phonetic Symbol has been transcribed into ASCII characters. It is important that, for the future KEC phone recognition tool to work, that the system be able to annotate the phones in a way consistent with the rest of corpus documentation. The original 340 Kiel labels extracted are condensed into 67 SAMPA labels, including diphthongs and a glottal stop. A selected sample of the condensed phone set can be found in the following table:

SAMPA	Kiel Modified SAMPA
aU	aU, aUE, aUO, aUa, aUaU„ aUo:
C	C, CQ, CS, Cd, Cf, Cg, Ch, Cj, Ck, Cs
i:6:	i:6„ i:66, i:6@, i:6E, i:6I, i:6i:, i:6y:, i:i:6:
t	t, tQ, tS, tb, td, tk, tn, tp, tq, ts, tz
v	v, v:, vb, vf, vh, vm, vz

(A phone followed by a colon in the Kiel Corpus denotes an elongated version of that sound.) The purpose of the models is to classify given speech into this set of 67 phone labels. A small number of the files were discarded due to data corruption. In total, there are 122510 phones in the data set used in the experiments.

3.2 Audio processing

Because I desired only audio features for each individual phone, extracting these several-millisecond-sized pieces audio from the longer sound files posed a challenge. At first I attempted to break the sound files into thousands of audio files based on phone boundary times. This requires a great deal of processing time. Another option was to extract spectrogram images for each phone and feed those directly into the models. Again, this would require an inefficient system of creating, storing, and

repeatedly accessing files from memory. Eventually, a solution was discovered to convert the entire sound file to its audio features, and then find the features between specific start and end times of each phone. The industry-standard Mel Frequency Cepstral Coefficients (MFCC) are computed from code developed by my research partner (Callihan 2018) and delta/delta delta from the librosa toolkit (McFee et al., 2015). Mel Cepstral Coefficients convert the audio signal to numeric matrices based on the energy intensities present at chosen frequencies, Delta and delta delta values, calculated from MFCCs, provide dynamic information about trajectory and acceleration of the speech signal. The Kiel Corpus files are sampled at their original 16 kHz rate. The MFCC features are computed with 12 coefficients on 25ms windows using a 10ms frame stride. A 15 ms overlap allows for a small amount of contextual information on either side of the window. The 12x12 feature vectors, while relatively small for this task, allow for lowered memory usage. The feature vectors are padded to maintain a uniform size.

3.3 The Models

I test phone classification with a number of architectures: a base model, transfer learning, transfer learning with SVM, and a gated recurrent unit (GRU cell). For the purpose of comparison and easy modification, the experiments center around a simple convolutional neural network architecture (CNN). The model with a GRU cell addition is implemented with the open source machine learning framework Tensorflow(Abadi et al., 2015), which allows for high levels of customization. Due to its higher-level ease of use, the Keras library(Chollet et al., 2015) with a Tensorflow backend is employed for all other model variations.

3.3.1 The base model

The base model is a simple convolutional neural network. It loosely based on the CIFAR-10 image recognition network, which is published as an example network by the Keras development team. A visualization of the network can be observed in Figure 3.1.

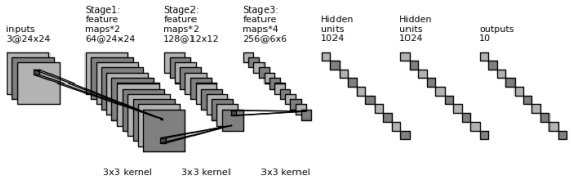

Figure 3.1: The CIFAR-10 architecture as presented in its paper. My model differs in that it only has one input channel (a 2D MFCC feature matrix rather than the three 2D RGB color channels of an image), there is no 256-unit layer, the 64 and 128-unit layers are doubled, and the kernel is 2x2 instead of 3x3. Source: (Abouelnaga et al., 2016)

I modify the architecture by doubling the convolutional layers before each pooling layer in order to improve feature representation, and thus, performance. This is shown in the following graphic:

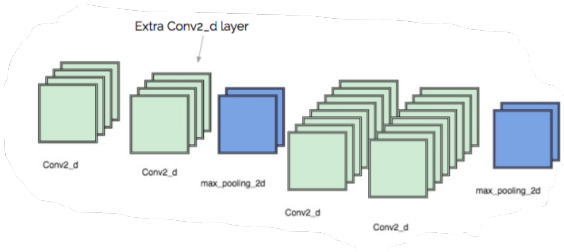

Figure 3.2: A representation of doubled convolutional layers.
Source: https://towardsdatascience.com/

Because there are only 144 features (12x12), the convolutional layers are of size 64 and 128, respectively. I use a kernel size of 2x2 within the convolutional layers and a pool size of 2x2 for the max pooling layers. The resulting features are then flattened into a one dimensional vector, and fed into a dense fully connected layer, and then again into a second dense layer. To decrease the likelihood of overfitting, there was 25% dropout rate after each pooling layer and the first fully connected layer. The activation function for the convolutional layers is a standard rectified linear unit (ReLU), which has proven ideal for use in acoustic models (Maas et al., 2013). The last dense layer of the network employs softmax. For the optimizer, I find that Adadelta works best for my purposes. The model's full architecture is specified in the table:

Layer (type)	Output Shape	Number of Parameters
2D Convolutional	(None, 11, 11, 64)	320
2D Convolutional	(None, 10, 10, 64)	16448
2D MaxPooling	(None, 5, 5, 64)	0
Dropout	(None, 5, 5, 64)	0
2D Convolutional	(None, 4, 4, 128)	32896
2D Convolutional	(None, 3, 3, 128)	65664
2D MaxPooling	(None, 1, 1, 128)	0
Dropout	(None, 1, 1, 128)	0
Flatten	(None, 128)	0
Dense	(None, 128)	16512
Dropout	(None, 128)	0
Dense	(None, 67)	8643

The CIFAR-10 architecture as presented in its paper. My model differs in that it only has one input channel (a 2D MFCC feature matrix rather than the three 2D RGB color channels of an image), there is no 256-unit layer, the 64 and 128-unit layers are doubled, and the kernel is 2x2 instead of 3x3.

For the run of the base model, the entire dataset of 122510 phone feature matrices is used. Mutually exclusive random training and test sets are created with a 60/40 split.

3.3.2 Transfer learning

The transfer learning experiments are an attempt to boost performance on the base model. My transfer models avoid typical "tabula rasa" random weight initialization through replacing these weights with those learned by a previous network. My particular style of transfer learning is inspired by Yosinski et al. (2014), who train models initialized on the weights of an identical model an unseen portion of the same dataset. Despite the simple nature of the setup, the research team reports a 2.1% point jump in performance. This appealed to me because it did not involve the potentially tedious preprocessing of another dataset in another language or the conceptualizing of another model type. For all of the models in these experiments, I use the same model architecture as my original base CNN. Due to relatively small input feature matrices and computational efficiency requirements, there are only four convolutional layers in the basic architecture (outline above in Section x). As a result, I choose to experiment with transferring pretrained weights from the first one, two, or three convolutional layers from a network A to another network B. Because transfer learning on the same data points would encourage overfitting, the Kiel dataset was split roughly in half yielding two smaller containing approximately 61000 phones each. In the experiments, the base model A with n convolutional layers (here, $n = 4$) is trained on the first portion of the data split. Weights from model

A are then stored as Numpy arrays (Walt et al., 2011) in a compressed binary file. Then, a model B is initialized with these pretrained weights for any convolutional layer between $n - 1$ and 0. When the transfer weights operate at $n = 2$, for example, the first two layers from model A are copied into model B. Finally, Model B is trained on the unseen, second portion of the data.

When transferring weights to neural network layers, one must decide whether these weights should be designated as trainable or frozen. Trainable layers, which can be updated along with the rest of the network during backpropagation, allow for fine tuning to the new data, while frozen layers do not change during training, which may improve efficiency and reduce overfitting. Yosinski et al. (2014), however, find a less of a performance increase when the layers were frozen than when they are not, presumably due to frozen weights being unable to adapt to new data. In their paper Transfer Learning for speech recognition on a budget, Kunze et al. (2017) confirm these findings: "We conclude that in terms of achieving small loss... freezing layers is not necessary." The results in the literature encouraged me to run almost all of the models in the transfer experiment with trainable weights. Still, I was interested in seeing the effect of frozen weights, so I created a model in which the weights were frozen for just the first layer. I assumed first layer features, which are general and often similar across networks, would be virtually interchangeable and lead to no significant loss in performance.

In summary, this section describes the creation of five separate models: model A trained on the first half of the Kiel Corpus data, models with trainable transfer weights copied into their first one, two, and three layers, and a model with frozen transfer weights copied into its first layer. The transfer models are all trained on the second half of the Kiel Corpus data.

3.3.3 CNN Features in Conjunction with a Support Vector Machine

The CNN-SVM model has a rather straightforward architecture. In this experiment, the CNN is used not for classification, but merely as a feature extractor. The full Kiel Corpus dataset is passed into the base CNN model (see Section 3.3.1). Highly-detailed features within the data, discovered by the convolutional and pooling layers, are the features that are passed into the SVM. Instead of having these features classified by the fully connected layer in the CNN, the output of the flatten layer is

extracted from the network, in a similar fashion to the work done by Sharif Razavian et al. (2014). The feature array is then fed into the SVM function provided by the Scikit-learn library. I use a radial basis function for the SVM, because, compared the linear SVM used in (CNN features CITE), it typically performs better when properly tuned. I employ a small grid search to find the best values for the regularization and γ parameters, which are found to be 1000 and .01, respectively. With this method, it is hoped that the SVM algorithm can find a plane which reasonably demarcates each of the 67 phone class boundaries within a vector space.

3.3.4 CNN Features in Conjunction with a Gated Recurrent Unit

The CNN-GRU system, like the CNN-SVM, primarily uses the convolutional part of the network as a feature extractor. As the only one of the experiments to include a recurrent neural network, this architecture is expected to pick up on patterns in the long-term temporal structure of the data. The final output for convolutional layers is fed into a gated recurrent unit, which cascades information from adjacent phones in both directions. The output of the GRU, in turn, is piped into a final fully connected layer, and class predictions are made. Like the base model, this model employs softmax as its activation function. One notable detail is that, in order to preserve temporal information, the data corresponding to individual phones was not randomized but rather left in the order spoken in the original utterance. This CNN-GRU portion of the project was developed with the guidance of my research partner, Ryan Callihan, who went on to use a modified version of this system in his Master's thesis on dialect recognition (Callihan, 2018)

4 Results and Discussion

Phone classification performance for all of the experiments fall into the accepted range for the task within the last thirty years. In addition, the models are reasonably efficient, achieving this performance in short amounts of time on my computer's hardware. Nevertheless, some of the findings are unexpected. In this section, I first report general observations and later remark on specific experiments in detail. Phone classification accuracy is used as the main metric for comparison across the models. All of the neural network-based models are run for 50 epochs.

The following table reports a comparison among all of the models:

Model Type	Accuracy	Training time (in minutes)
svm	60.14%	2 minutes
full set	71.45%	54 minutes
network a	**72.28%**	**34 minutes**
third conv	68.67%	28 minutes
second conv	68.45%	27 minutes
first conv	68.24%	28 minutes
frozen weights	67.93%	17 minutes
with GRU	70.12%	156 minutes

As can be seen, the highest performance is achieved with the simple CNN model A that utilizes half of the original dataset. The base model of the same architecture, trained on the entire dataset, may offer a better view into this particular CNN's capabilities. The CNN-GRU hybrid, while performing well, does not outperform the other networks due to its incorporation of contextual information, as previously hypothesized.

4.1 Transfer learning results

When initializing a network with pretrained weights, one expects that the new network, already having latent "experience" with data, is more robust and resistant to falsely modeling new data. In my experiment, unfortunately, this is not the case. While there is a slight trend of performance improvement as higher-level weights were transferred, as predicted by Yosinski et al. (2014), the transfer models never reached the performance achieved by the the base model A. In the graph in Figure 4.1, the accuracy of the 2-layer transfer model is compared with model A over 50 epochs. We see that accuracy for the transfer model begins considerably lower, as expected. However, the accuracy transfer model is never able to surpass that of the original model at any point in time.

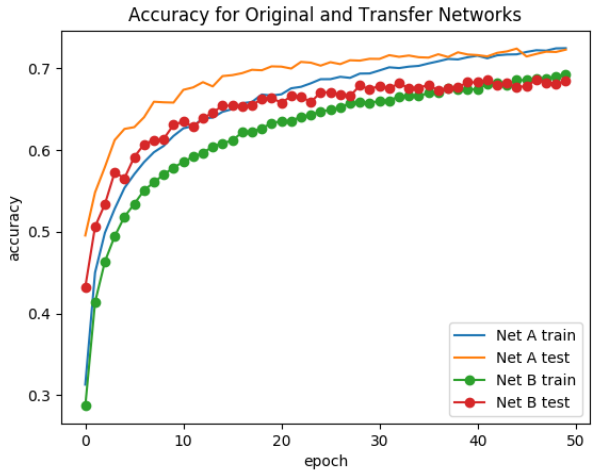

Figure 4.1: The accuracy achieved by the base model and a model with transferred weights

In the literature surveyed, freezing transfer weights does not lead to any improvement in performance; performance is even slightly reduced. This is confirmed in the graph in Figure 4.2, which compares a network having its first layer with frozen transfer weights to the original model A:

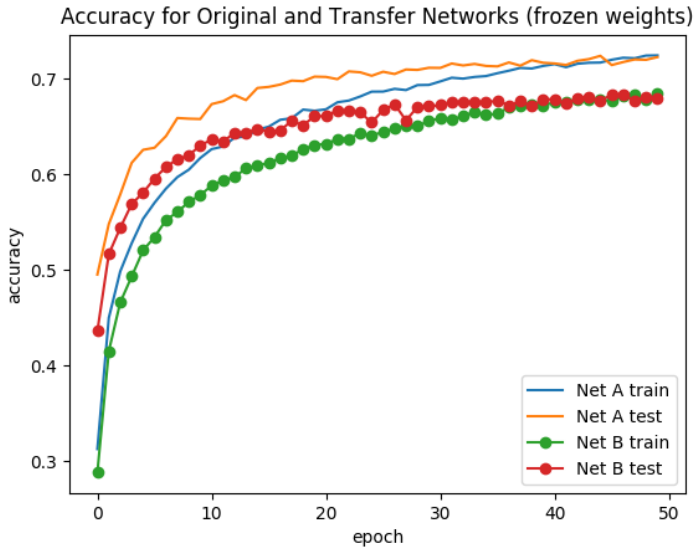

Figure 4.2: The accuracy achieved by the base model and a model with transferred weights that have been frozen

The explanation for these findings may be that the halved dataset was too small, thus not allowing for suitable generalization to be passed to the transfer weights. Another concern of mine was that the nature of the read German speech data at hand did not contain enough variance in its features. Perhaps if model A had also been presented with spontaneous

German data or data from another language or dialect, the weights would have exhibited more robustness against unseen data. An interesting variation on this experiment would have been to use transfer weights from a much more complex ASR architecture like Deep Speech. (I looked into this, and could not find a suitable transfer learning interface for their program at the time of writing). In the end, transferring weights into neural networks may simply not be promising method from which to expect high jumps in performance. Jarrett et al. (2009) note that a combination of "random convolutional filters, rectification, pooling, and normalization" can work almost as well as learned features, and this is mirrored in my own results.

4.2 CNN-SVM Results

The CNN-SVM model is both the least time consuming as well as the lowest performing, at 60.14%. Because data among the phone classes is not evenly distributed, I was curious to see other metrics such as, recall, precision, and F1 score for this model. The featured table includes a classification report for the CNN-SVM across the 67 classes:

Phone	Precision	Recall	F1-score	Support
y:	0.62	0.47	0.53	17
t	0.65	0.80	0.72	292
E:6	1.00	0.20	0.33	5
b	0.67	0.66	0.67	74
n	0.59	0.80	0.68	333
p	0.33	0.12	0.18	24
a:	0.39	0.50	0.44	76
2:	1.00	0.50	0.67	10
v	0.77	0.55	0.64	65
m	0.62	0.44	0.51	89
U6	0.77	0.87	0.82	31
k	0.73	0.70	0.72	114
f	0.92	0.89	0.90	87
u:6	0.67	0.13	0.22	15
2:6	0.92	0.66	0.77	35
z	0.43	0.30	0.35	10
Z	0.48	0.53	0.51	117
E6	0.00	0.00	0.00	2
I	0.45	0.60	0.52	116
~	0.87	0.75	0.80	55
a	0.38	0.40	0.39	172
x	0.00	0.00	0.00	14
Q	0.30	0.15	0.20	75
E:	0.57	0.46	0.50	57
o:	0.69	0.36	0.47	25
OU	0.26	0.23	0.25	86
e:I	0.65	0.66	0.65	56
d	0.79	0.62	0.70	24
l	0.42	0.48	0.45	193
aU	0.48	0.36	0.41	58
q	0.78	0.87	0.82	352
e:	0.18	0.15	0.16	27
Oa	0.73	0.80	0.76	10
a6	0.89	0.94	0.91	17
u:	0.00	0.00	0.00	3
w	0.65	0.42	0.51	31
@	0.73	0.62	0.67	13
aI	0.56	0.20	0.29	25
h	0.15	0.14	0.15	21
<p>	0.83	0.56	0.67	9
9	0.45	0.31	0.37	32
S	0.75	0.33	0.46	9
?	0.00	0.00	0.00	8
y:6	0.43	0.37	0.40	84
i:	0.17	0.06	0.08	18
OY	0.61	0.51	0.56	70
N	0.33	0.34	0.33	50
E	0.89	0.79	0.83	70
Y6	0.92	0.94	0.93	141
6	0.73	0.73	0.73	11
A	0.43	0.22	0.29	46
j	0.00	0.00	0.00	9
i:6	0.40	0.67	0.50	3
g	0.40	0.14	0.21	14
avg / total	0.59	0.60	0.59	3400

We can observe that there is a positive correlation between the number of instances of each phone class and the SVM's ability to place an instance in its correct class. Because the kernel used for my particular SVM is nonlinear, it runs in quadratic time, and is thus inefficient at processing large amounts of data. As I run the algorithm many times to find the best γ and regularization parameters, I use a mere 180 speech-annotation file pairs, amounting to 3400 phone feature arrays, in order to keep execution time low. The small amount of data used may be the reason for the

relatively low performance. With more compute resources, this issue could surely be solved.

4.3 CNN-GRU Results

Accuracy from the CNN-GRU achieves 70.12% , higher than the transfer and SVM models, yet lower than the simple CNN base model. This is somewhat striking, because, due to the temporal nature of speech, one might expect that the gated recurrent unit's attention to structure would allow it to significantly outperform other models. Not only did the CNN-GRU not exhibit this performance, but it takes approximately twice as long to train as the simple CNN. An inspection of the CNN-GRU's accuracy curve in Figure 4.3 shows a typical pattern.

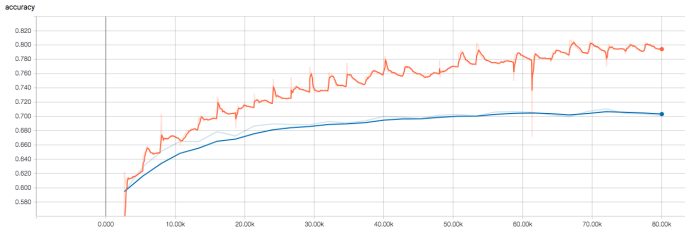

Figure 4.3: The accuracy achieved by the CNN-GRU model

One theory to explain the performance may be that, as (Arnold & Tomaschek, 2017) observe, phones are only a loose theoretical unit that to not capture the intricacies of speech, and so any system based upon them is inherently flawed. Indeed, current state-of-the-art systems like Deep Speech bypass the phone unit entirely. A modification that may have boosted performance may have been to condense phone classes further. After all, (Raphael, 1972) show that crucial phone identifying features may sometimes be omitted in spontaneous speech; so mistaking a *p* for a *b* may be unavoidable with limited context, and indeed, evade even human recognition capabilities. An even more drastic measure may have been to narrow down predictions by guessing mode or place of articulation before identifying the phone itself, as proposed by (Scanlon et al., 2007). Nevertheless, the CNN-GRU, along with the other models implemented for this thesis, perform suitably in accordance with related literature. Research into phone recognition remains ongoing, and still offers the benefit of improving the field of ASR as a whole.

5 Conclusion

The models presented in this offer unique approaches for the task of phone classification. The experiments performed serve as an invaluable opportunity to gain an intuition about the complex problem at hand. I achieve my research goal of conceptualizing, testing, and achieving good performance on numerous resource-efficient models. While model test accuracies behave differently than expected, with a simple CNN architecture surpassing both transfer learning models and CNN-RNN hybrid, all perform at between 67.93% and 72.28% accuracy. This puts them into the range of state-of-the-art results reported for phone recognition in the last twenty years (Lopes & Perdigao, 2011) The CNN-SVM, with 60.14% accuracy, approaches this range, while also providing interesting insights into possible later experiments. This research project is a small contribution to the field of automatic speech recognition, and a successful step toward an automated pipeline for annotation of the Karl Eberhards Corpus.

6 Acknowledgements

I would especially like to thank my research partner and colleague, Ryan Callihan. He was instrumental in explaining concepts and the theory behind sound processing in the early weeks of this project, as well as for his assistance with the GRU model development. I would also like to extend thanks to the project leader Dr. Fabian Tomaschek, who employed me as his research assistant at the Quantitative Phonetics department at the University of Tuebingen. Lastly, I thank Prof. Kurt Eberle, who agreed to act as my official advisor for the thesis.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., & Penn, G. (2012). Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics, speech and sig-*

nal processing (icassp), 2012 ieee international conference on (pp. 4277–4280).

Abouelnaga, Y., Ali, O. S., Rady, H., & Moustafa, M. (2016). Cifar-10: Knn-based ensemble of classifiers. In *Computational science and computational intelligence (csci), 2016 international conference on* (pp. 1192–1195).

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... others (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning* (pp. 173–182).

Arnold, D., & Tomaschek, F. (2017). *The karl eberhards corpus of spontaneously spoken southern german in dialogues-audio and articulatory recordings.* Institut für Deutsche Sprache, Bibliothek.

Callihan, R. (2018). *Dialect recognition using audio signals and deep learning* (Unpublished master's thesis). University of Tuebingen.

Chollet, F., et al. (2015). *Keras.* `https://keras.io`.

Clark, J., & Yallop, C. (1994). *C.(1994). an introduction to phonetics and phonology.* Cambridge: Blackwell.

Cooke, M., Green, P., Josifovski, L., & Vizinho, A. (2001). Robust automatic speech recognition with missing and unreliable acoustic data. *Speech communication*, *34*(3), 267–285.

Crystal, D. (2011). *A dictionary of linguistics and phonetics* (Vol. 30). John Wiley & Sons.

Golipour, L., & O'Shaughnessy, D. (2009). Context-independent phoneme recognition using a k-nearest neighbour classification approach. In *Acoustics, speech and signal processing, 2009. icassp 2009. ieee international conference on* (pp. 1341–1344).

Gruhn, R., & Hamerich, S. (2011, June 30). *Grammar and template-based speech recognition of spoken utterances.* Google Patents. (US Patent App. 12/634,302)

Haykin, S. (1994). *Neural networks: a comprehensive foundation.* Prentice Hall PTR.

Jarrett, K., Kavukcuoglu, K., LeCun, Y., et al. (2009). What is the best multi-stage architecture for object recognition? In *Computer vision, 2009 ieee 12th international conference on* (pp. 2146–2153).

Johnson, K. (2004). Massive reduction in conversational american english. In *Spontaneous speech: Data and analysis. proceedings of the 1st session of the 10th international symposium* (pp. 29–54).

Kipp, A., Wesenick, M.-B., & Schiel, F. (1997). Pronuncation modeling applied to automatic segmentation of spontaneous speech. In *Fifth european conference on speech communication and technology.*

Kohler, K. J. (1996). Labelled data bank of spoken standard german: the kiel corpus of read/spontaneous speech. In *Spoken language, 1996. icslp 96. proceedings., fourth international conference on* (Vol. 3, pp. 1938–1941).

Kunze, J., Kirsch, L., Kurenkov, I., Krug, A., Johannsmeier, J., & Stober, S. (2017). Transfer learning for speech recognition on a budget. *arXiv preprint arXiv:1706.00290*.

Lamel, L. F., & Gauvain, J.-L. (1993). High performance speaker-independent phone recognition using cdhmm. In *Third european conference on speech communication and technology.*

Li, J. (2008). *Soft margin estimation for automatic speech recognition* (Unpublished doctoral dissertation). Georgia Institute of Technology.

Lopes, C., & Perdigao, F. (2011). Phoneme recognition on the timit database. In *Speech technologies.* InTech.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (Vol. 30, p. 3).

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference* (pp. 18–25).

Mohamed, A.-r., Dahl, G., & Hinton, G. (2009). Deep belief networks for phone recognition. In *Nips workshop on deep learning for speech recognition and related applications* (Vol. 1, p. 39).

Mohamed, A.-r., Sainath, T. N., Dahl, G., Ramabhadran, B., Hinton, G. E., & Picheny, M. A. (2011). Deep belief networks using discriminative features for phone recognition. In *Acoustics, speech and signal processing (icassp), 2011 ieee international conference on* (pp. 5060–5063).

Ostendorf, M. (1999). Moving beyond the 'beads-on-a-string'model of speech. In *Proc. ieee asru workshop* (pp. 79–84).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Raphael, L. J. (1972). Preceding vowel duration as a cue to the perception of the voicing characteristic of word-final consonants in american english. *The Journal of the Acoustical Society of America*, *51*(4B), 1296–1303.

Rapp, S. (1995). Automatic phonemic transcription and linguistic annotation from known text with hidden markov models. an aligner for german.

Robinson, A. J. (1994). An application of recurrent nets to phone probability estimation. *IEEE transactions on Neural Networks*, *5*(2), 298–305.

Scanlon, P., Ellis, D. P., & Reilly, R. B. (2007). Using broad phonetic group experts for improved speech recognition. *IEEE transactions on audio, speech, and language processing*, *15*(3), 803–812.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 815–823).

Schweitzer, A., & Lewandowski, N. (2013). Convergence of articulation rate in spontaneous speech. In *Interspeech* (pp. 525–529).

Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 806–813).

Walt, S. v. d., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22–30.

Wang, D., & Zheng, T. F. (2015). Transfer learning for speech and language processing. *arXiv preprint arXiv:1511.06066*.

Xu, Y., Kong, Q., Huang, Q., Wang, W., & Plumbley, M. D. (2017). Convolutional gated recurrent neural network incorporating spatial features for audio tagging. In *Neural networks (ijcnn), 2017 international joint conference on* (pp. 3461–3466).

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems* (pp. 3320–3328).

Young, S. J. (1992). The general use of tying in phoneme-based hmm speech recognisers. In *icassp* (pp. 569–572).

Zue, V., Seneff, S., & Glass, J. (1990). Speech database development at mit: Timit and beyond. *Speech communication*, *9*(4), 351–356.